OpusTools and Parallel Corpus Diagnostics

Aulamo, Mikko

European Language Resources Association (ELRA)
2020-05-17

Aulamo , M , Sulubacak , U , Virpioja , S & Tiedemann , J 2020 , OpusTools and Parallel Corpus Diagnostics . in N Calzolari , F Béchet , P Blache , K Choukri , C Cieri , T Declerck , S Goggi , H Isahara , B Maegaard , J Mariani , H Mazo , A Moreno , J Odijk & S Piperidis (eds) , Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020) . European Language Resources Association (ELRA) , Paris , pp. 3782-3789 , Language Resources and Evaluation Conference , 11/05/2020 .

http://hdl.handle.net/10138/320206

Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 3782–3789
Marseille, 11–16 May 2020

OpusTools and Parallel Corpus Diagnostics

Mikko Aulamo, Umut Sulubacak, Sami Virpioja, Jörg Tiedemann
Department of Digital Humanities
University of Helsinki, Helsinki / Finland
{mikko.aulamo, umut.sulubacak, sami.virpioja, jorg.tiedemann}@helsinki.fi

Abstract

This paper introduces OpusTools, a package for downloading and processing parallel corpora included in the OPUS corpus collection. The package implements tools for accessing compressed data in their archived release format and make it possible to easily convert between common formats. OpusTools also includes tools for language identification and data filtering as well as tools for importing data from various sources into the OPUS format. We show the use of these tools in parallel corpus creation and data diagnostics. The latter is especially useful for the identification of potential problems and errors in the extensive data set. Using these tools, we can now monitor the validity of data sets and improve the overall quality and consistency of the data collection.

Keywords: Corpus (Creation, Annotation, etc.); Machine Translation; Tools, Systems, Applications

1. Introduction

OPUS (Tiedemann, 2012) is the biggest collection of openly available parallel corpora. The collection has been growing constantly over the years and is widely used in work on machine translation and cross-linguistic research. Currently it contains 57 released corpora covering over 700 languages and language variants creating more than 70,000 bitexts in the sense of aligned language pairs across all corpora in the collection. The size and popularity of OPUS makes it necessary to build an efficient infrastructure that enables the various users to obtain and access the data and this paper introduces two packages that provide tools for that purpose. The goal of those packages is it to make it easy to download, convert and process the data included in OPUS from the command line or from applications using the library implementing those tools. The two packages refer to a Python library with command-line tools and a complementary Perl module, both provided as open source and with permissive licenses.

In the sections below, we introduce the tools and their basic use and also discuss how we applied those tools to create new data sets and to run systematic diagnostics of the entire data base. With the availability of the OpusTools it is now possible to run careful sanity checks on the extensive data sets to verify validity of encoding, to find broken links and structures and to identify other issues with the data.

2. Characteristics of OPUS

OPUS includes parallel corpora from a wide variety of sources. Each of them comes with their own peculiarities and the properties can differ substantially depending on the original data and their distribution. The philosophy in OPUS is to keep markup and annotation as much as possible but to unify the essential data format to make access to parallel data as transparent as possible. This means that corpus data is converted to standalone (schema-free) XML that keeps original markup but consistently adds essential markup that is necessary for alignment and further linguistic processing. Alignment is stored as standoff annotation in XCES Align format (for sentence alignment) and "Moses format" (for word alignment). Using this principle, data can be kept apart from alignment annotation, which enables efficient implementation and storage of massively parallel data and also allows alternative alignments if necessary.

Figure 1 shows an example of standoff annotation used in OPUS for specifying links between sentences. Each sentence alignment file may include an arbitrary number of `linkGrp` elements to align documents from a data collection. Documents are specified using a path relative to the XML root of the OPUS sub-corpus and `link` elements provide the sentence alignments by sets of sentence ID's that are separated by semicolon. Creating an alternative alignment is simply done by creating a new sentence alignment file and no further modifications need to be done with the original corpus data. Note that sentence alignment is bilingual as shown in the example. However, standoff annotation makes it possible to align massively parallel data sets across all language pairs without duplicating any of the linked data files. Furthermore, there can be alternative corpus files with different levels of annotation without the need of re-aligning those alternative files. Figure 2 shows examples of such annotated files all aligned in the same way with the standoff sentence alignment stored in external files. More details about the data structures in OPUS can be found on the OPUS Wiki.[1]

Another principle in OPUS is to provide the data in other common formats to make them easily accessible for a wide range of applications. Those data formats are, however, just generated from the underlying XML-based encoding, which serves as the master copy of each corpus. Users of OPUS data are typically not aware of those principles and download the data format that most suits their needs.

The idea of OpusTools is now to unify the access to master data in XML and to the other generated formats by providing essential libraries and command-line tools to retrieve and convert corpus data. They also provide convenient tools for basic filtering and random access in archived data in their compressed form that is used for distributing the data. The latter is especially important as the size of some cor-

[1] `http://opus.nlpl.eu/trac/wiki/DataFormats`

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cesAlign PUBLIC
                  "-//CES//DTD XML cesAlign//EN" "">
<cesAlign version="1.0">
<linkGrp targType="s"
         fromDoc="en/0/1089124/4995691.xml.gz"
         toDoc="fr/0/1089124/4588599.xml.gz">
<link id="SL0" xtargets="1;1" overlap="0.331" />
<link id="SL1" xtargets="2 3;2" overlap="0.560" />
<link id="SL2" xtargets="4;" />
<link id="SL3" xtargets="5 6;3" overlap="0.854" />
<link id="SL4" xtargets="7 8 9;4" overlap="0.699" />
<link id="SL5" xtargets="10 11;5" overlap="0.776" />
...
```

Figure 1: An example of standoff sentence alignment in XCES Align format. The `linkGrp` element specifies the document pairs that are aligned and links between individual sentences are given in the `link` elements. The optional `overlap` attributes in this example refer to time overlap ratios that are used as a feature in subtitle alignment.

pora is extensive in such a way that it is demanding for common file systems to handle the data in raw, uncompressed form. For example, the latest OpenSubtitles corpus contains roughly 3.7 million individual documents across 67 languages with alignment in over 3,600 bitexts. One of the latest additions, JW300 covers 380 languages in over 46,000 bitexts. Altogether there are over 9.2 million individual documents only in the latest releases of all corpora and this number is doubled by the different pre-processing types that are provided, raw text and tokenised corpora that are partially annotated with additional linguistic information. Furthermore, bitexts are released in native XML format (see Figure 2), plain text format and translation memory exchange (TMX) format. The releases currently occupy a total of 5.9 TB of space in compressed format.
The numbers above illustrate the need for proper infrastructures and efficient tools to manage the various data sets. This is the motivation for implementing the freely available OPUS tools described below. They create a convenient library and tool box for downloading, extracting and converting data from the OPUS collection. Additionally, they help to run systematic diagnostics on the collection to identify errors and problems in the data sets. Below, we will first present the two packages and their functionality. Thereafter we provide information about their use in creating new data sets and, finally, we report on the application of OPUS tools for diagnostic studies and sanity checks.

3. The OpusTools Package

The OpusTools package is a toolkit for downloading and managing parallel corpora data from OPUS. The package consists of a Python library and related command-line scripts. Additionally, there is a Perl package for creating new data sets and accessing parallel data.

3.1. Command-Line Tools

The OpusTools package includes five Python 3 based command-line scripts: `opus_read`, `opus_express`, `opus_cat`, `opus_get` and `opus_langid`.[2] The scripts allow downloading OPUS data, outputting the data in specific formats, extracting training, development and test sets from the data, and more. Figure 3 shows an overview of the scripts.

opus_read is a script for downloading parallel corpora and converting them to desired formats. OPUS corpora contain XCES format alignment files that point to two XML sentence files in different languages. The XCES alignment format links the sentences in source files to the sentences in target files using sentence ID's. The sentence files in OPUS corpora are compressed into ZIP archives and `opus_read` makes it convenient to read the data directly from the compressed files. `opus_read` parses a given alignment file and produces an output in one of four formats: normal, moses, TMX or XCES links. `opus_read` first tries to read the OPUS files from local directories. If the required files are not found, the tool offers an option to download them. The sentence files can be downloaded in raw, tokenised or parsed format.
`opus_read` includes basic filters for removing unwanted sentence pairs before creating the output file. Non-alignments, where the source or target segment is empty, can be left out. Alternatively, a certain number of source and target segments can be specified, e.g. it is possible to include only one-to-one alignments in the output. Some corpora include an attribute score for each sentence pair. For example, sentence pairs in the OpenSubtitles corpus have overlap scores that indicate to what degree the time stamps of the two segments overlap. `opus_read` is able to filter out sentence pairs that do not cross a given attribute score threshold. Furthermore, segment pairs can be removed based on language identification confidence scores. Language labels and confidence scores can be added to sentence XML files with `opus_langid` script.

opus_express is a script built on `opus_read` that can extract ready-to-use training, development, and test sets for a language pair from one or more OPUS corpora. The procedure first fills the specified quota of sentences for the test set, then continues with the same for the development set, and dumps the rest into the training set. The script can optionally pre-shuffle the data before splitting, or conversely, mark and preserve document boundaries across the splits for document-level models. `opus_express` also includes an option to utilise attribute scores such as overlap values as extracted by `opus_read` in its quality-awareness toggle, which prioritises higher-confidence sentence pairs surpassing a configurable threshold to be sorted into the test and development sets.

opus_cat is used for reading monolingual corpora from OPUS or single files within those corpora. The files can be printed out in XML format or they can be converted into plain text. `opus_cat` is useful for manually inspecting the domain or the quality of a single corpus because it is able to read files directly from the ZIP archives in OPUS corpora.

opus_get is a script for downloading parallel corpus files from OPUS. Before downloading, corpora can be searched and listed by their name, source language and target language. For example, one can download files for a specific language pair in a single corpus, all language pair files in

[2] `https://github.com/Helsinki-NLP/OpusTools`

Raw XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<document>
  <CHAPTER ID="1">
    <P id="1">
      <s id="1">Resumption of the session</s>
    </P>
    <SPEAKER ID="1" NAME="President">
      <P id="2">
        <s id="2">I declare resumed the session of the European Parliament adjourned on Thursday, 14 June 2001.</s>
      </P>
```

Tokenized (annotated) XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<document><CHAPTER ID="1"><P id="1">
<s id="1">
 <chunk type="NP" id="c-1">
  <w hun="NN" tree="NN" lem="resumption" pos="NN" id="w1.1">Resumption</w>
 </chunk>
 <chunk type="PP" id="c-2">
  <w hun="IN" tree="IN" lem="of" pos="IN" id="w1.2">of</w>
 </chunk>
 <chunk type="NP" id="c-3">
  <w hun="DT" tree="DT" lem="the" pos="DT" id="w1.3">the</w>
  <w hun="NN" tree="NN" lem="session" pos="NN" id="w1.4">session</w>
 </chunk>
</s>
```

UD Parsed XML format:

```
<?xml version="1.0" encoding="utf-8"?>
<document>
 <CHAPTER ID="1">
  <P id="1">
   <s id="1">
    <w xpos="NOUN" head="0" feats="Number=Sing" upos="NOUN" lemma="Resumption" id="1.1" deprel="root">Resumption</w>
    <w xpos="ADP" head="1.4" upos="ADP" lemma="of" id="1.2" deprel="case">of</w>
    <w xpos="DET" head="1.4" feats="Definite=Def|PronType=Art" upos="DET" lemma="the" id="1.3" deprel="det">the</w>
    <w xpos="NOUN" head="1.1" feats="Number=Sing" upos="NOUN" misc="SpaceAfter=No" lemma="session" id="1.4"
       deprel="nmod">session</w>
   </s>
```

Figure 2: Examples of XML encoded data in OPUS. Various kinds of annotations can be added without destroying the sentence alignment, which is stored as standoff annotation of links between sentence ID's.

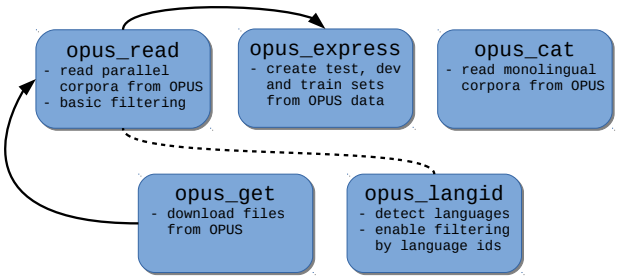

Figure 3: The five Python based OpusTools scripts. Each of the scripts can be used individually. `opus_express` is build on `opus_read` and `opus_read` uses `opus_get` to download OPUS files. `opus_langid` has to be applied to sentence files to enable language id filtering for `opus_read`.

a single corpus or all files for a specific language in the whole OPUS. `opus_read` uses `opus_get` for automatically downloading the requested corpus files.

opus_langid is used for adding language identification labels and confidence scores for each sentence in a given XML sentence file. Language identification is carried out with two off-the-shelf tools: pycld2[3], the Python bindings for Compact Language Detector 2[4] and `langid.py` (Lui and Baldwin, 2012). `opus_langid` has to be applied to sentence XML files before `opus_read` can filter sentence pairs by their language labels. Figure 4 shows an example of a sentence file that has been processed with `opus_langid`.

3.2. The OpusTools Python Library

In addition to being command-line scripts, `opus_read`, `opus_cat`, `opus_get` and `opus_langid` are associated with Python modules that can be imported and used within one's own scripts. The modules provide the same functionality as the command-line tools and also more detailed data managing control by the use of submodules and functions. All Python code is written in Python 3.

OpusRead module can be initialised with parameters that correspond with the flags given to `opus_read` and is used for downloading and converting corpus files from OPUS. Internally, `OpusRead` uses XML parsing modules from the `parse` sublibrary included in the OpusTools Python package. The sublibrary contains modules for parsing XCES alignment files and sentences files. The

[3] `https://github.com/aboSamoor/pycld2`
[4] `https://github.com/CLD2Owners/cld2`

```
<?xml version="1.0" encoding="utf-8"?>
<text>
 <p id="1">
  <s cld2="en" cld2conf="0.99" id="s1.1" langid="en" langidconf="1.0">
   Statement of Government Policy by the Prime Minister, Mr Ingvar Carlsson, at the Opening of the Swedish
   Parliament on Tuesday, 4 October, 1988.
  </s>
 </p>

 <p id="2">
  <s cld2="en" cld2conf="0.98" id="s2.1" langid="en" langidconf="1.0">
   Your Majesties, Your Royal Highnesses, Mr Speaker, Members of the Swedish Parliament.
  </s>
 </p>
```

Figure 4: Example of a sentence file, where language labels and confidence scores have been added to sentence tags.

`AlignmentParser` module parses a given XCES link file and initialises `SentenceParser` modules for parsing the sentences files. `AlignmentParser` outputs single sentence pair segments, while `SentenceParser` outputs single sentences from either side of the alignment. `LinksAlignmentParser` can be used in the case that only the XCES links are needed and sentence file parsing can be skipped. For sentence parsing, there is also an alternative module `ExhaustiveSentenceParser`, which is more robust than `SentenceParser` but slightly slower when parsing only a small portion of a large corpus. Each of the modules in `parse` sublibrary can be individually imported into a Python script and used to extract single sentences, sentence pairs or XCES links.

OpusCat is the Python module used by the `opus_cat` script and both have the same functionality of reading monolingual sentence files from OPUS. `OpusCat` utilises a modified version of the `SentenceParser` module: when reading single sentence files, the sentence parsing process does not need to follow an order specified in an alignment file, and the `SentenceParser` in `OpusCat` simply outputs each sentence in a file. Both `OpusCat` and `SentenceParser` can be imported as Python modules to have detailed control over reading monolingual files.

OpusGet module powers the `opus_get` script with corpora downloading capabilities. By importing the module in Python code, one is able to receive detailed information about OPUS corpora within Python data structures. This information includes number of alignment pairs, number of documents, number of tokens and size in kilobytes among other items.

OpusLangid module has the same functionality as the `opus_langid` script: adding language labels and language identification confidence scores to XML sentence files. Additionally, `OpusLangid` contains `LanguageIdAdder` class, which can be used for obtaining language labels and identification confidence scores from both pycld2 and `langid.py` for a plain text sentence with single function call.

3.3. The OpusTools Perl Module

A complementary package of OPUS tools is provided as a Perl module available with a permissive MIT license.[5] It includes command-line tools that are handy especially for the creation of new data sets but also in general for quickly accessing data in different formats. Some of the functionality is now superseded by the implementations in the Python library described above, and we will here focus on the tools that support additional use cases. Those tools mainly fall into the following three categories:

Conversion tools: Tools that can be used to import and export data in different file formats and data markup. The main purpose is to import new data sets in OPUS and to create data files that are released with different formats.

Alignment tools: Sentence and word alignment can be used in various ways and these tools provide some convenient operations on top of aligned bitexts.

Other processing tools: This category includes tools for annotation and indexing.

In the first category, we have import tools such as `moses2opus`, `tmx2opus` and `xml2opus`. Export scripts include `opus2moses`, `tmx2moses`, `opus2text` and `opus2multi`.

xml2opus is a simple script that adds sentence boundaries to arbitrary XML data. Sentence boundary detection is done using the tools released with the Europarl parallel corpus (Koehn, 2005) and packaged in the Perl module `Lingua::Sentence`. Additional tools based on UD treebank classifiers will be integrated in the future. Inline tags that add markup within sentences are not supported at the moment.

moses2opus reads aligned plain text files as commonly used in machine translation with aligned sentences on the same line.[6] The tool converts the data into simple standalone XML for the corpus data and the XCES Align format for standoff sentence alignment as it is used within OPUS. Currently, only bilingual input is supported. Plain text files do not contain sentence boundaries but still may contain sentence alignments that are not one-to-one. Therefore, `moses2opus` adds sentence markup using `Lingua::Sentence` and adjusts the standoff sentence alignment accordingly. The script also supports splitting bitexts into smaller parts. Empty lines in source and

[5] `https://github.com/Helsinki-NLP/OpusTools-perl`

[6] The name comes from the Moses package that popularised the format.

target language can be used to indicate document boundaries. Furthermore, a corpus can be split into equally sized portions using a length threshold for the maximum number of translation units included in one part.

tmx2opus converts translation memories in TMX format into OPUS XML. The tool adds sentence boundaries in the same way as `moses2opus` does. It also allows to pipe several TMX files through the conversion tool and it is able to merge information in case of overlapping sentences that are covered in several translation units. This is handy when processing data that comes as different bitexts but covering the same content. Hence, only unique sentences are stored in the resulting OPUS XML for each language even though they appear in different translation units with alignments to various languages. `tmx2opus` can also process translation memories with more than two languages in a translation unit, and it will produce bilingual sentence alignment files for all language pairs, as they are necessary in OPUS. Furthermore, it is also possible to split data into smaller portions similar to what `moses2opus` does. Properties from TMX files can also be copied to the converted data in order to keep additional meta data. The application of `tmx2opus` for the creation of the imported ParaCrawl corpus in OPUS is described in Section 4..

Export scripts mainly perform data conversion in the opposite direction. `opus2moses` and `opus2text` convert OPUS XML data to plain text and they are mostly obsolete and superseded by the implementation of the Python package introduced earlier. `tmx2moses` is a convenient script to extract aligned sentences from arbitrary TMX files and it is not restricted to OPUS data.

opus2multi is a tool that can create multiparallel data sets from OPUS corpora. In OPUS, all data sets are aligned bilingually but in some cases one would like to have an alignment that spans more than two languages. For this, `opus2multi` can help to join bilingual sentence alignments and to extract links across a larger number of languages. The tool operates on standoff sentence alignment files and makes use of a pivot language to construct translation units across all given languages. For this, it expands partially overlapping sentence alignments until all languages are covered without further conflicts in the resulting translation unit (i.e. no remaining overlaps with other units). The result of that process is sentence alignment files that are (for convenience) printed bilingually using the XCES Align format, which can then be further processed using OpusTools to extract the actual alignment pairs. There is also an option to control the maximum size of a translation unit (in number of sentences in one language) as the size can grow without limits in the expansion process. An experimental feature of including intralingual links for further transitive mapping is also included. This is handy for data sets like OpenSubtitles in which alternative subtitle files may be used for linking between different languages.

Alignment tools in the OpusTools package help to process sentence alignments in their standoff annotation format. `opus-swap-align` simply swaps the alignment direction. OPUS only provides alignments in one direction (as they are symmetric anyway) but sometimes it is convenient to have access to the links in the other direction as well. `opus-merge-align` combines sentence alignment files and deletes duplicates if there are any. `opus-split-align` splits sentence alignment files into separate files with one per alignment group, i.e. aligned document. Finally, `opus-pivoting` makes it possible to create transitive sentence alignment between two languages using a pivot language and links to the pivot language. This is convenient for corpora that come with bitexts that do not cover all language pairs but only align to a specific language like English. Assuming that there is substantial overlap between the bitexts, let us say $A \rightarrow P$ and $B \rightarrow P$, `opus-pivoting` extracts links between sentences in A and B, creating a new bitext $A \rightarrow B$. Section 4. illustrates the use by the example of the creation of MultiParaCrawl. Finally, another alignment tool, `opus-pt2dice`, extracts rough probabilistic bilingual dictionaries from phrase-translation-tables created from word alignment and using SMT tools coming out of the Moses toolbox. Those dictionaries use some heuristics to filter the data and the tool also creates additional Dice scores as a symmetrised alignment value out of the conditional translation probabilities included in the original phrase tables, which is useful for bilingual lexicon extraction (Smadja et al., 1996).

Other tools: The last tool category contains additional data processing tools such as `opus-udpipe` and `opus-index`. The former implements a wrapper around UDPipe (Straka and Straková, 2017) to annotate OPUS data and to store the result in OPUS-conforming XML. OpusTools can use pre-trained models coming from LINDAT.[7] Last but not least, `opus-index` is a tool for indexing OPUS corpora using the Corpus Work Bench (CWB) (Evert and Hardie, 2011). It creates all import files and runs the encoder if available to create multiparallel corpora to be queried using the CWB search engine.

4. ParaCrawl and MultiParaCrawl

In this section, we would like to showcase the import of the ParaCrawl data to demonstrate the use of OpusTools. The ParaCrawl corpus[8] has been extracted by crawling the Web and applying a complex document and sentence alignment pipeline based on the Bitextor package (Esplà-Gomis, 2009). The current release v5.0 covers 24 European languages and the project provides automatically cleaned bitexts for languages aligned to English. The size ranges from 100,000 translation units (Maltese-English) to over 50 million units (French-English) and the data files are distributed in plain text or TMX format. While there are a few bonus language pairs that also include two bitexts not including English, the majority of the collection is bilingually aligned with English content.

The goal of the integration of ParaCrawl in OPUS is to make the data available via the native OPUS format and to also exhaustively cover all language pairs included in the collection. For those purposes, the previously introduced

[7] `https://lindat.mff.cuni.cz`

[8] `https://paracrawl.eu`

language	files	tokens	sentences	bg	cs	da	de	el	es	et	fi	fr	ga	hr	hu	it	lt	lv	mt	nl	pl	pt	ro	sk	sl	sv
bg	1	57.7M	2.6M		0.5M	0.4M	0.7M	0.4M	0.7M	0.3M	0.4M	0.8M	96.6k	0.3M	0.3M	0.5M	0.3M	0.2M	68.0k	0.4M	0.4M	0.5M	0.4M	0.4M	0.2M	0.4M
cs	1	119.0M	5.3M	0.5M		0.8M	1.4M	0.6M	1.2M	0.4M	0.6M	1.3M	0.1M	0.4M	0.6M	1.2M	0.3M	0.3M	79.1k	0.9M	1.0M	1.0M	0.6M	0.8M	0.3M	0.7M
da	1	108.3M	4.7M	0.4M	0.8M		1.4M	0.6M	1.4M	0.4M	0.8M	1.4M	0.1M	0.4M	0.5M	1.3M	0.3M	0.3M	88.3k	1.3M	0.9M	1.2M	0.5M	0.5M	0.3M	1.3M
de	1	909.7M	38.3M	0.7M	1.4M	1.4M		0.8M	7.0M	0.4M	0.8M	8.1M	0.1M	0.5M	0.8M	6.0M	0.4M	0.3M	82.8k	3.1M	1.8M	3.6M	0.7M	0.6M	0.4M	1.4M
el	1	94.9M	3.8M	0.4M	0.6M	0.6M	0.8M		1.0M	0.2M	0.5M	1.0M	0.1M	0.3M	0.4M	0.9M	0.3M	0.2M	76.1k	0.7M	0.6M	0.9M	0.5M	0.4M	0.3M	0.6M
es	1	961.5M	38.7M	0.7M	1.3M	1.5M	7.1M	1.0M		0.4M	0.9M	9.9M	0.1M	0.5M	0.8M	6.8M	0.4M	0.3M	78.2k	2.9M	1.8M	6.0M	0.9M	0.6M	0.3M	1.4M
et	1	26.5M	1.4M	0.3M	0.4M	0.4M	0.4M	0.2M	0.4M		0.4M	0.4M	95.1k	0.2M	0.3M	0.3M	0.3M	0.2M	81.2k	0.3M	0.3M	0.3M	0.3M	0.2M	0.2M	0.4M
fi	1	54.4M	3.2M	0.4M	0.6M	0.8M	0.8M	0.5M	0.9M	0.4M		1.0M	0.1M	0.3M	0.5M	0.8M	0.3M	0.3M	80.7k	0.8M	0.7M	0.8M	0.5M	0.4M	0.2M	1.2M
fr	1	1.3G	51.1M	0.8M	1.4M	1.4M	8.3M	1.0M	10.1M	0.4M	1.0M		0.1M	0.5M	0.8M	7.1M	0.4M	0.3M	82.3k	3.4M	1.8M	4.6M	0.9M	0.6M	0.4M	1.4M
ga	1	24.8M	0.8M	97.6k	0.1M	0.1M	0.1M	0.1M	0.1M	96.4k	0.1M	0.1M		67.5k	0.1M	0.1M	78.0k	75.7k	54.7k	99.4k	98.2k	0.1M	75.6k	0.1M	76.7k	96.5k
hr	1	43.2M	1.9M	0.3M	0.5M	0.4M	0.5M	0.3M	0.5M	0.2M	0.3M	0.5M	68.2k		0.3M	0.6M	0.2M	0.2M	50.6k	0.4M	0.4M	0.4M	0.3M	0.3M	0.3M	0.3M
hu	1	107.0M	4.1M	0.3M	0.7M	0.5M	0.8M	0.4M	0.8M	0.3M	0.5M	0.9M	0.1M	0.3M		0.8M	0.3M	0.3M	76.4k	0.6M	0.7M	0.6M	0.6M	0.5M	0.3M	0.5M
it	1	562.3M	22.0M	0.5M	1.2M	1.3M	6.1M	1.0M	7.0M	0.4M	0.8M	7.2M	0.1M	0.6M	0.8M		0.4M	0.3M	91.4k	2.6M	1.7M	3.9M	0.9M	0.6M	0.4M	1.3M
lt	1	25.6M	1.3M	0.3M	0.3M	0.3M	0.4M	0.3M	0.4M	0.3M	0.4M	0.4M	79.0k	0.2M	0.3M	0.4M		0.3M	73.4k	0.4M	0.4M	0.4M	0.3M	0.3M	0.2M	0.4M
lv	1	22.5M	1.1M	0.2M	0.3M	0.3M	0.3M	0.2M	0.3M	0.2M	0.3M	0.3M	76.5k	0.2M	0.3M	0.3M	0.3M		66.9k	0.3M	0.3M	0.3M	0.2M	0.3M	0.2M	0.3M
mt	1	4.2M	0.2M	68.4k	79.5k	88.8k	83.3k	76.5k	78.7k	81.7k	81.1k	82.9k	55.0k	50.8k	76.8k	92.0k	73.8k	67.2k		85.7k	86.2k	87.2k	68.7k	82.6k	71.5k	86.5k
nl	1	237.9M	10.6M	0.4M	0.9M	1.3M	3.1M	0.8M	3.0M	0.3M	0.8M	3.5M	0.1M	0.4M	0.6M	2.7M	0.4M	0.3M	86.4k		1.2M	2.2M	0.6M	0.5M	0.3M	1.3M
pl	1	144.8M	6.7M	0.4M	1.1M	0.9M	1.9M	0.6M	1.9M	0.3M	0.7M	1.8M	99.3k	0.4M	0.7M	1.8M	0.4M	0.3M	86.8k	1.2M		1.5M	0.7M	0.6M	0.3M	1.0M
pt	1	320.1M	13.5M	0.5M	1.0M	1.2M	3.6M	0.9M	6.1M	0.3M	0.8M	4.7M	0.1M	0.4M	0.7M	4.0M	0.4M	0.3M	87.9k	2.2M	1.6M		0.7M	0.5M	0.3M	1.2M
ro	1	65.7M	2.9M	0.4M	0.6M	0.5M	0.7M	0.5M	0.9M	0.3M	0.5M	0.9M	76.4k	0.3M	0.6M	0.9M	0.3M	0.2M	69.2k	0.6M	0.7M	0.7M		0.4M	0.3M	0.6M
sk	1	41.6M	2.1M	0.4M	0.8M	0.5M	0.6M	0.4M	0.6M	0.2M	0.4M	0.6M	0.1M	0.3M	0.5M	0.6M	0.3M	0.3M	83.1k	0.5M	0.6M	0.5M	0.4M		0.4M	0.5M
sl	1	31.8M	1.5M	0.2M	0.3M	0.3M	0.4M	0.3M	0.3M	0.2M	0.2M	0.4M	77.5k	0.3M	0.3M	0.4M	0.2M	0.2M	71.9k	0.3M	0.4M	0.3M	0.3M	0.4M		0.3M
sv	1	131.5M	6.1M	0.4M	0.7M	1.3M	1.4M	0.6M	1.5M	0.4M	1.2M	1.4M	97.5k	0.3M	0.5M	1.4M	0.4M	0.3M	87.1k	1.3M	1.0M	1.2M	0.6M	0.5M	0.3M	

Figure 5: Statistics from the MultiParaCrawl corpus - a multilingual extension of ParaCrawl via pivot alignment through English. The upper-right triangle gives the size in terms of sentence alignments in plain text format, and the lower-left triangle shows the size of the extracted TMX files in terms of unique translation units per language pair.

tools `tmx2opus` and `opus-pivoting` become handy. `tmx2opus` is not only useful for extracting the alignments from the original TMX source, but it also provides the functionality to add sentence boundary markup and to reduce redundancy between the different bitexts. Using the unique option of `tmx2opus` reduces the size of the English portion of the corpus (i.e. 252 million separately aligned English sentences in 23 bitexts) to less than 60% of the original data. At the same time, the uniqueness feature also enables to build a multiparallel corpus by pivoting on the links to English in the newly created unique set of sentences. For that, `opus-pivoting` can be used as explained earlier. Using this procedure, 253 additional bitexts could be created with sizes up to 10 million sentence-aligned translation units. Figure 5 summarises the non-English bitexts in MultiParaCrawl.

5. Parallel Corpus Diagnostics

Our diagnostic routine for the OPUS collection uses the `opus_read` command line utility (described in Section 3.1.) to retrieve aligned plain text data for a particular language pair in a given corpus. In order to do this, `opus_read` parses the native XML-formatted data to generate the requested subset of data, and then performs a conversion to plain text format. During this process, the diagnostic routine listens for any errors that might arise, and logs them to compile a diagnostic report for later analysis. We perform this procedure systematically for every pair of languages available under each of the OPUS corpora.[9]

For our diagnostics, we exploit the full granularity provided by OPUS by gathering separate readings for different corpora that compose bitexts, and also keeping regional variants of languages separate rather than conflating them. To carry out this kind of exhaustive analysis, we ran a total 87,948 CPU array jobs in parallel, with runtimes varying between 1 second and 5.2 hours, and each job using between 4 and 128 GBs of memory. In total, the entire diagnostic analysis took approximately 1000 hours of computation, averaging 18.2 hours per corpus. While the granularity of our analysis will be internally useful for pinpointing anomalies in OPUS to facilitate repairs, we also collate our data to generate corpus-wide figures, which we report and discuss in this section.

5.1. Error Analysis

The "diagnoses" logged in our report lists the causes of each retrieval error, which provides us with a means to reliably locate and fix them. Collating all diagnoses, the results reveal that while 37 of the corpora are completely error-free, data retrieval stalled for at least one language pair for the remaining 18 corpora. The abundance of retrieval errors in these corpora vary from a tiny fraction to the entire corpus (shown in Figure 8). The vast majority of these errors stem from ill-formed XML data with invalid tokens (96.2%) or mismatched tags (3.5%). Our partial checks so far suggest that these can be attributed to minor conversion errors such as unescaped special characters and XML entities occurring in the original data prior to the import to OPUS. Another very small portion of the errors (0.3%) indicates missing data files in the main file system where OPUS is hosted, which likely indicates copy errors, and remains to be investigated further.

5.2. Corpus-Wide Statistics

In addition to cataloguing the data retrieval issues, our diagnostic procedure also calculates some basic quantitative statistics, such as the reported computational costs for data retrieval, and various measurements on the data retrieved per corpus, language, and language pair. Our corresponding statistical analyses did not reveal noteworthy trends or outliers for the most part, except for some measures that hinted at the relative variances and noise levels in data across corpora. In Figures 6 and 7, we report

[9] We did not perform diagnostics on the two most recent additions to OPUS: infopankki and MultiParaCrawl.

distributions of two measures over the set of available language pairs for each corpus: the average number of sentence pairs (or, more accurately, translation units), and the average sentence length in characters, respectively. Both measures were visualised using box-and-whisker plots to emphasise distributional differences, where the endpoints

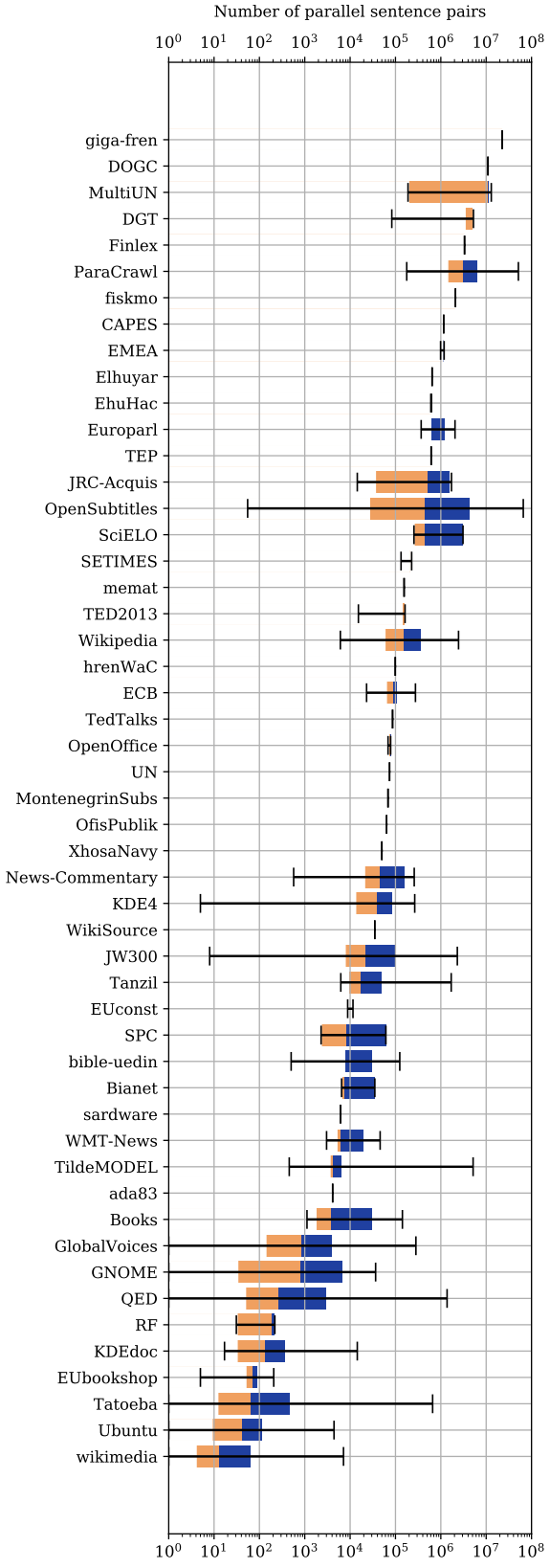

Figure 6: Distribution of the number of retrievable parallel sentence pairs over the set of available language pairs for each corpus.

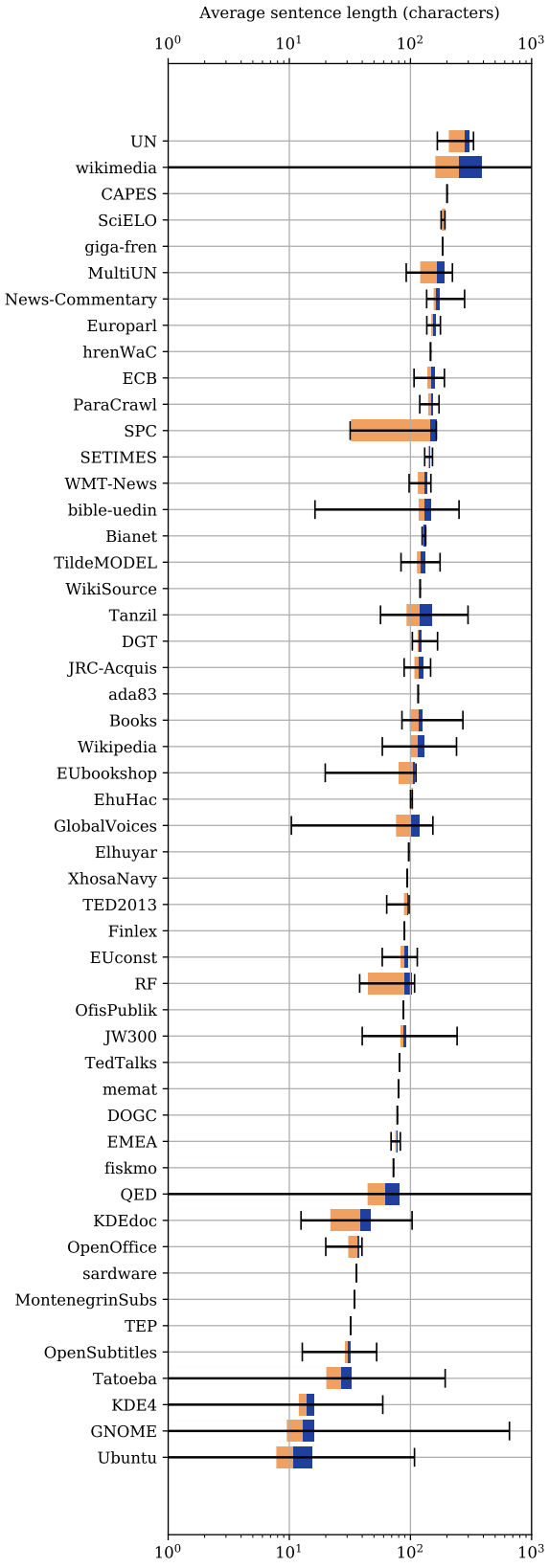

Figure 7: Distribution of average sentence lengths (in characters) over sets of available language pairs for each corpus.

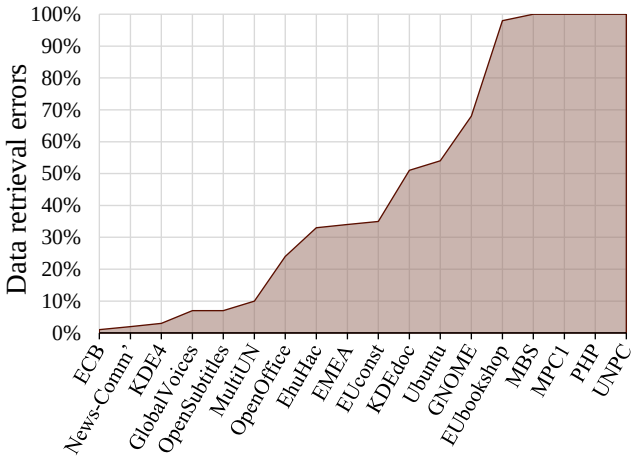

Figure 8: Percentages of language pairs in OPUS corpora for which data retrieval with OPUS tools returns errors. Error-free corpora were omitted from graph.

show the lowest and highest values[10], and the two halves of the box represent the second and third quartiles of values, separated by the median.

One of the most striking details from Figure 6 is the contrast between variances. More than a third of the corpora show very little to no variance across language pairs, which implies fully multiparallel data, while others like JW300 and OpenSubtitles conversely demonstrate very high variance, where the difference in the sizes of available data may span several orders of magnitude. Looking specifically at the first quartiles, some corpora such as QED and Tatoeba seem to have a significant portion of language pairs containing very few translation units, possibly indicating high language detection or sentence alignment noise. In Figure 7, the first quartile appears to have a similar relative range for some corpora, meaning that sentences contain only a few characters on average for some of the available language pairs. It is likely not a coincidence that these cases mostly correspond to corpora that were compiled from naturally noisy data. Furthermore, the smallest and largest median values in Figure 7 point to exceptionally short and exceptionally long "typical" sentences in the corresponding corpora, which may indicate a strong contrast in text segmentation, or distinctly different data domains. For example, the three corpora with the lowest medians include translations of computer software, while documents from United Nations yield the highest median length.

6. Conclusions and Future Work

In this paper, we introduce OpusTools, an open-source package of libraries and command-line tools for efficient and convenient access to parallel corpora in the extensive OPUS data collection. The package implements tools for downloading, converting, filtering and processing parallel data sets and makes it easy to access compressed and archived files from the collection. It also provides a python library for programmatic access to the data making it easy to incorporate data processing in the development of other tools. Furthermore, we present tools for data conversion and alignment that can be applied when preparing new data sets from various sources. We demonstrate their use with the example of the recently added MultiParaCrawl corpus that extends the original data set with pivot-based alignments between all language pairs contributing to the growing coverage of the OPUS database.

Though keeping a collection as large as OPUS perfectly robust is quite challenging, troubleshooting will be easier and faster with the diagnostics fully charted out. All in all, while data retrieval errors comprise clear action points, the statistical analyses rather seem to suggest a notable qualitative and quantitative diversity among OPUS corpora, with trends seemingly within expectations, and edge cases that can be attributed to noise in the original data. Our intention is to resolve all issues around data retrieval, so that using OPUS tools will be a smooth experience for all users, and also to streamline our routine as a diagnostic tool, which would become a standard part of the process of expanding OPUS with new corpora.

[10]Uncapped endpoints indicate extrema beyond the limits of the x-axis.

Acknowledgments

This work is part of the FoTran project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement № 771113), as well as the MeMAD project, funded by the European Union's Horizon 2020 Research and Innovation Programme (grant agreement № 780069).

7. Bibliographical References

Evert, S. and Hardie, A. (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 conference*, University of Birmingham, UK.

Smadja, F., McKeown, K. R., and Hatzivassiloglou, V. (1996). Translating collocations for bilingual lexicons: A statistical approach. *Computational Linguistics*, 22(1):1–38.

8. Language Resource References

Esplà-Gomis, Miquel. (2009). *Bitextor: a Free/Open-source Software to Harvest Translation Memories from Multilingual Websites*.

Koehn, Philipp. (2005). *Europarl: A Parallel Corpus for Statistical Machine Translation*. AAMT.

Lui, Marco and Baldwin, Timothy. (2012). `langid.py`: *An Off-the-shelf Language Identification Tool*. Association for Computational Linguistics.

Straka, Milan and Straková, Jana. (2017). *Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe*. Association for Computational Linguistics.

Tiedemann, Jörg. (2012). *Parallel Data, Tools and Interfaces in OPUS*. European Language Resources Association (ELRA).